Interactive 3-D Graphics for the Apple II

An understanding of the theory of perspective enables you to represent three-dimensional objects on a two-dimensional screen.

Andrew Pickholtz
3613 Glenbrook Rd.
Fairfax, VA 22031

In the present generation of computers, no other form of output rivals the popularity of the video terminal with its two-dimensional visual representation of data. This article will examine ways of making this two-dimensional output represent the three-dimensional real world. Techniques of showing perspective play an important role in making video output look three-dimensional. In this article, I will look briefly at the concept of perspective and then consider some techniques of achieving perspective in computer graphics. I will then present some program listings in BASIC and Pascal that show how to use these techniques in high-level languages.

About the Author

Andrew Pickholtz wrote this article while a senior at W. T. Woodson High School in Fairfax, Virginia. He is now a student at Harvard University. In summers, he has worked for Ferox Microsystems Inc. and at the IBM T. J. Watson Research Center.

Acknowledgments

The author would like to thank the Department of Electrical Engineering and Computer Science, George Washington University, for the use of its word processor and Apple II computer systems. He would also like to thank Dr. and Mrs. R. Pickholtz for persuading him to write this article.

Ways of Representing Three Dimensions

People tried to portray the visual world on a flat screen long before the creation of the modern computer, and draftsmen today use several different methods of representing three-dimensional objects on paper: the orthogonal, the oblique, the isometric, and the perspective methods.

A computer can as easily produce a perspective drawing as an oblique or isometric drawing.

An orthogonal projection of an object is simply the "side view" of that object (see figure 1). "Side view" is in quotes because, as will later become clear, this representation is not exactly what the human eye would see if it were looking at the object; that is, this "view" is not a perspective projection.

Orthogonal representations of objects customarily give three projections: one from the top, one from the front, and one from the right-hand side. Each "view" gives information about a pair of axes; the "top view" gives information about the x-y pair, the "front view" about the x-z pair, and the "right-hand side view" about the y-z pair. Unfortunately, the untrained eye is reluctant to form a three-dimensional image from the three detached and seemingly independent illustrations used in orthogonal representation.

Oblique and isometric drawings (see figures 2 and 3, respectively) portray an object in a more realistic manner. Both the oblique and isometric representations depict a three-dimensional object in one illustration by fixing the axes in relation to the horizontal. In oblique pictorial, lines parallel to the z axis are vertical, lines parallel to the x axis are horizontal, and lines parallel to the y axis are consistently drawn at the same angle in relation to the horizontal. The axes in isometric pictorial are likewise fixed in relation to the paper.

The Perspective Method

While oblique and isometric representations are superior to orthogonal,

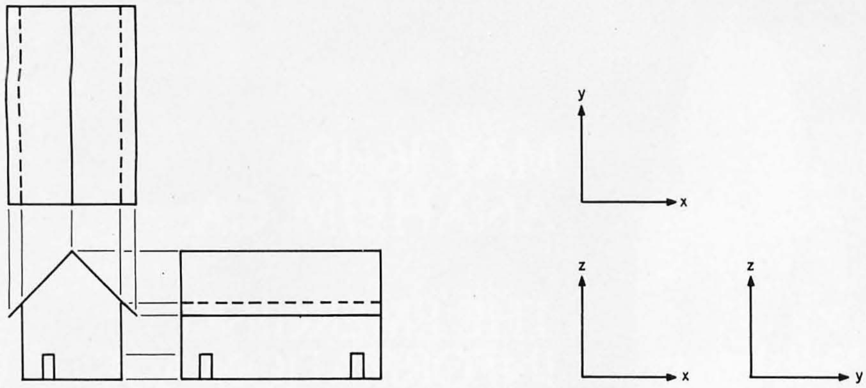

Figure 1: *An orthogonal representation of a house. An orthogonal drawing drops perpendiculars from each point on the object to three mutually perpendicular planes. Hidden edges are customarily drawn as dotted lines.*

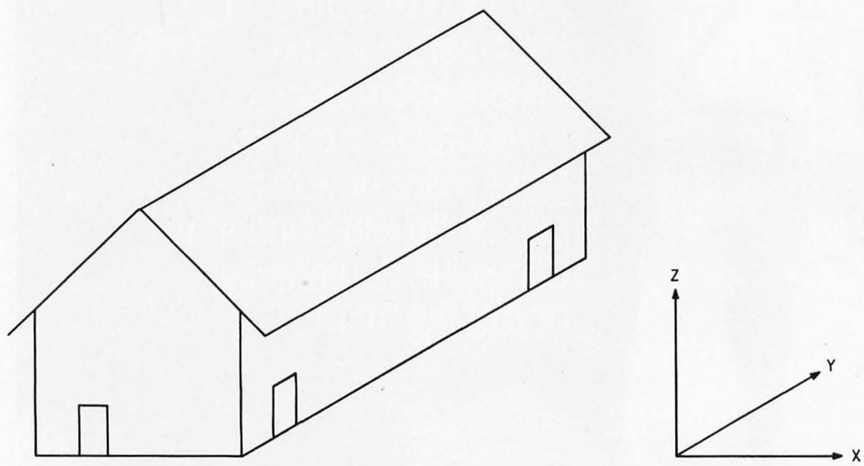

Figure 2: *An oblique representation of a house. An oblique drawing portrays three dimensions by drawing lines parallel to the third axis at a consistent angle to the horizontal, in this case at 30 degrees.*

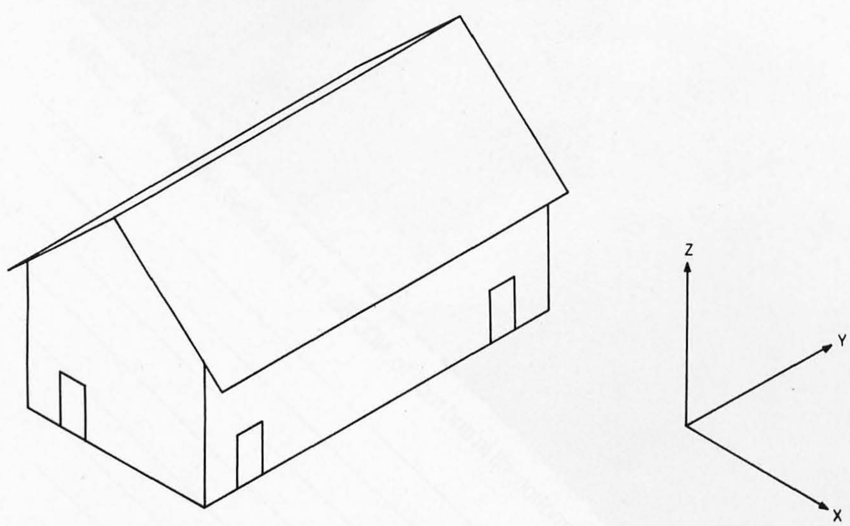

Figure 3: *An isometric representation of a house. Like an oblique representation, an isometric one draws lines that are parallel in three dimensions as parallels in two. The isometric method, however, offsets two axes from the horizontal.*

perspective pictorial is the only truly accurate method of illustrating an object. Two Florentine architects, Filippo Brunelleschi and Leon Battista Alberti, developed the ideas of perspective in the fifteenth century. Although many artists before them had noticed that objects in the distance appear smaller than objects in the foreground, Brunelleschi and Alberti were the first to accurately represent the apparent diminution of objects as they recede from the observer. Many other Italian artists and some Flemish artists had also experimented with perspective; however, their methods were empirical while Brunelleschi and Alberti worked with a geometric system. In fact, Alberti had written several papers on mathematics, and in 1435 wrote the first treatise on painting that dealt with the theory of art rather than just the techniques.

What makes perspective drawings superior to oblique and isometric is that perspective displays objects in the distance as smaller than objects that are closer; the rear door in figure 4, for example, is smaller than the front door. Perspective drawing also represents lines that are parallel in three dimensions as convergent on the picture plane. Thus, the axes in perspective drawings are always directed toward vanishing points. The x-axis and y-axis vanishing points in figure 4 lie on the horizon; an object in the distance would, as the eye expects, appear extremely small.

Figures 5a-5c illustrate another interesting fact about perspective: while the oblique (figure 5a) and isometric (figure 5b) representations of a wire-frame cube appear to spontaneously reverse in orientation, the perspective representation (figure 5c) does not. What prevents the spontaneous reversal in the perspective representation is that one of the perceived orientations of the perspective cube is erroneous; that is, it does not look "natural."

Although oblique and isometric drawings are not truly realistic, draftsmen use these two techniques more often than perspective. They do this for two reasons. First, oblique and isometric drawings conveniently

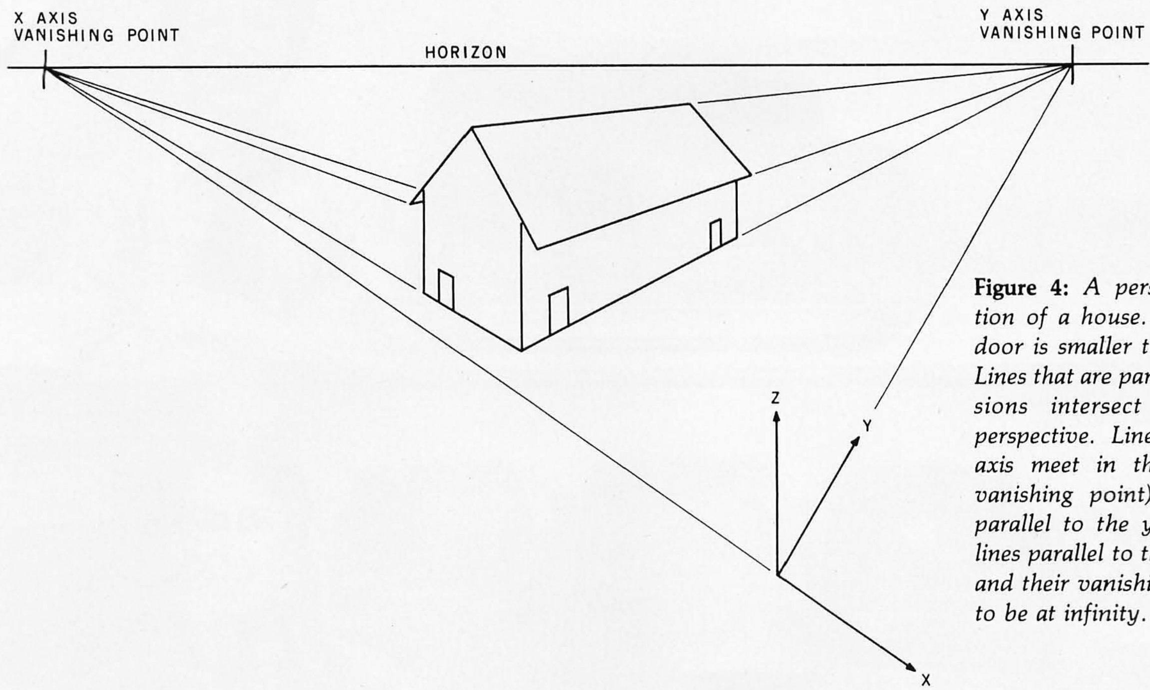

Figure 4: *A perspective representation of a house. Note that the rear door is smaller than the front door. Lines that are parallel in three dimensions intersect when drawn in perspective. Lines parallel to the x axis meet in the distance (at the vanishing point), and so do lines parallel to the y axis. In this case, lines parallel to the z axis are vertical and their vanishing point is assumed to be at infinity.*

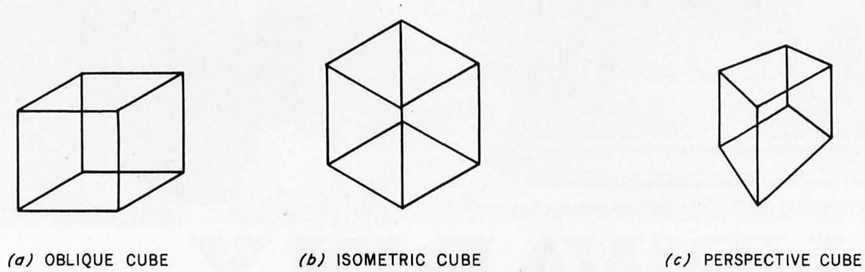

Figure 5: *Three representations of a cube. Figure 5a is an oblique representation, figure 5b is isometric, and figure 5c is a perspective. Both the oblique and the isometric representations appear to reverse in orientation spontaneously. The perspective does not.*

permit finding the measurements of an object by simply measuring the representation; second, drawing perspective is much more difficult. For a computer, however, it is just as easy to produce a perspective drawing as to produce an oblique or isometric drawing. Furthermore, as the object becomes more complex, the difference in speed between computer-drawn perspective and computer-drawn isometric becomes negligible.

Describing a Three-Dimensional Object

It is impossible to produce a perspective pictorial of an object without a description of the object. A good representation of the object can usually be achieved by assuming that the object is composed of a finite number of planar polygons. If the object is significantly curved, an adequate representation requires many polygons.

Figure 6 illustrates a data structure that describes a three-dimensional object. Each of the polygons, which can be called faces, is composed of edges. Each edge is composed of two vertices that are specified by three Cartesian coordinates. Each face also has several characteristics: color, texture, transmittance, glossiness, and reflectance. The edge shared by two faces is the intersection of their two sets of coordinates.

It is easier to represent an object if we assume that the object has clear faces. This simplification avoids the difficult problem of discovering hidden lines. Figure 7 shows the simpler data structure that this assumption permits to represent the wire-frame object previously represented in figure 6.

Specifying an Arbitrary Three-Dimensional View

We can think of a perspective pictorial of a three-dimensional scene as a view that a one-eyed pilot would see when looking through an empty picture frame (see figure 8). The picture frame is understood to lie in the picture plane. As the figure shows, the pilot's line of sight is defined to be the normal (perpendicular) to the picture plane that passes through the pilot's eye. The lines connecting the object with the pilot's eye are called projectors. The perspective pictorial is the intersection of the projectors and the picture plane.

Three general types of changes would affect the pilot's view of the scene: a change in the distance between the picture plane and the pilot's eye, a change in position of the aircraft, or a rotation of the airplane. If the picture plane is moved closer to the pilot's eye, the view would appear smaller in comparison to the picture frame. Likewise, if the picture frame is moved further away from the pilot's eye, the view would appear larger since the tetrahedral angle that the picture frame subtends (marks off) would be smaller. Thus, in order to specify any three-dimensional

VERTICES $V_1\ (X_1, Y_1, Z_1)\quad V_2\ (X_2, Y_2, Z_2) \cdots V_{NV}\ (X_{NV}, Y_{NV}, Z_{NV})$

EDGES $E_1 = V_1, V_4 \quad E_2 = V_5, V_9 \quad E_3 = V_2 V_{104} \cdots E_{NE} = V_{16} V_3$

FACES $F_1 = E_1 E_3 \cdots E_{118} \quad F_2 = E_{209} E_5 E_{19} \cdots F_{NF} = E_3 E_{52} \cdots E_{129}$

COLOR_1	$\text{COLOR}_2 \cdots$	COLOR_{NF}
TEXTURE_1	$\text{TEXTURE}_2 \cdots$	TEXTURE_{NF}
GLOSSINESS_1	$\text{GLOSSINESS}_2 \cdots$	GLOSSINESS_{NF}
⋮	⋮	⋮
REFLECTANCE_1	$\text{REFLECTANCE}_2 \cdots$	REFLECTANCE_{NF}

OBJECT $O_1 = F_1 F_2 F_3 \cdots F_{NF}$

Figure 6: *A data structure representing an object. The representation assumes that the object is composed of a finite number of polygons (also called faces). Each face has several characteristics (color, etc.) and is determined by the edges that it contains. Each edge is specified by its endpoints, which are the vertices of the object. Finally, the coordinates of each vertex must be specified. The object represented here is hypothetical.*

VERTICES $V_1\ (X_1, Y_1, Z_1)\quad V_2\ (X_2, Y_2, Z_2) \cdots V_{NV}\ (X_{NV}, Y_{NV}, Z_{NV})$

TRAILS $T_1 = V_1 V_3 V_8 \cdots V_{186}$

$T_2 = V_9 V_{12} V_{136} \cdots V_1$

$T_3 = V_{13} V_{12} \cdots V_{13}$

⋮

$T_{NT} = V_4 V_{11} \cdots V_{22}$

OBJECT $O_1 = T_1 T_2 T_3 \cdots T_{NT}$

Figure 7: *A data structure representing a wire-frame object. This data structure assumes that the object has transparent faces. The object is composed of trails. Each trail is defined by the vertices that it contains. Each vertex is specified by its coordinates. The data structure shown represents a hypothetical object.*

view, we must know the distance between the picture plane and the observer's eye, the position of the observer, and the angular position of the observer's line of sight.

Figure 9 illustrates the linear and angular position of an observer. Three Cartesian coordinates specify the location of the observer. The coordinate axes that specify the observer's location are the same axes used to specify the vertices of the object. Describing a unique line of sight requires three angles—pitch, bank, and heading. A change in pitch is a rotation about the wings. A change in bank, or roll, is a rotation about the fuselage. And a change in heading, or yaw, is a rotation about a vertical line passing through the pilot; in other words, the heading is the compass direction of the airplane.

Since rotation is not a commutative operation—one in which a change in order will not change the results—we must declare an order of precedence for pitch, bank, and heading. The most physically appealing order is heading, pitch, and then bank. Using that order, we can determine a line of sight by first rotating a unit vector parallel to the y axis about the z axis in an amount specified by the heading. Next, we should rotate the new vector about the new position of the wings by an amount specified by the pitch. And finally, we should

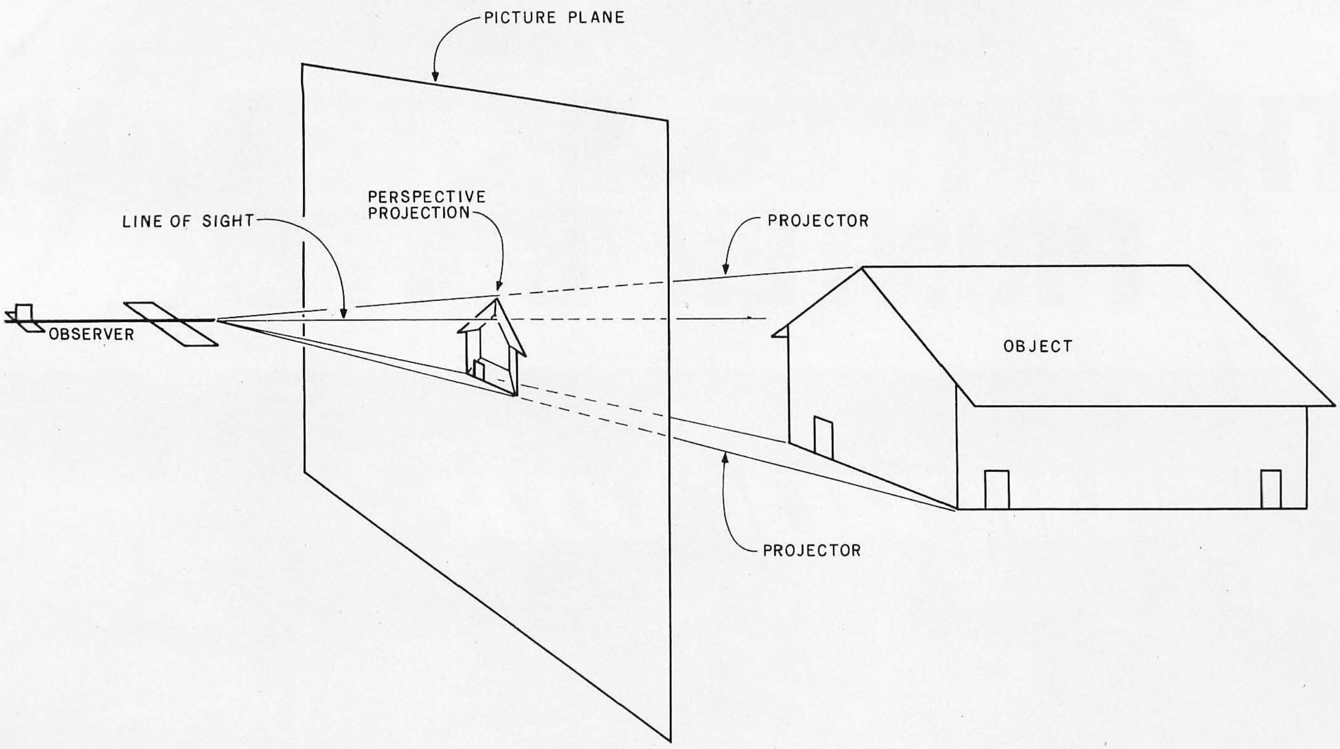

Figure 8: *Perspective projection of an object. The observer's line of sight is normal to the picture plane. The projectors of an object are the lines connecting the object to the observer's eye. The perspective projection consists of the intersections of the projectors and the picture plane. The distance between the observer and the picture plane controls the size of the perspective projection; the farther the picture plane is from the observer, the larger the projection.*

rotate this new vector about the newly positioned fuselage in an amount specified by the bank.

Solving for the Standard Position

Later, we will see that the computations required to create a perspective projection can be greatly simplified by translating and rotating the coordinate system so that the observer is at the origin, with the line of sight aligned with the positive y axis, and the wings aligned with the x axis. When the observer is in this standard position, the pitch, bank, and heading are defined to be zero. We can move the observer to the standard position only if we likewise move the three-dimensional scene so that the observer's view remains unchanged.

Assume that the observer is at the location (X_V, Y_V, Z_V) and has pitch, bank, and heading p, b, and h, respectively. Translating the observer to (0,0,0) and a point $Q(X,Y,Z)$ to $(X-X_V, Y-Y_V, Z-Z_V)$ does not alter

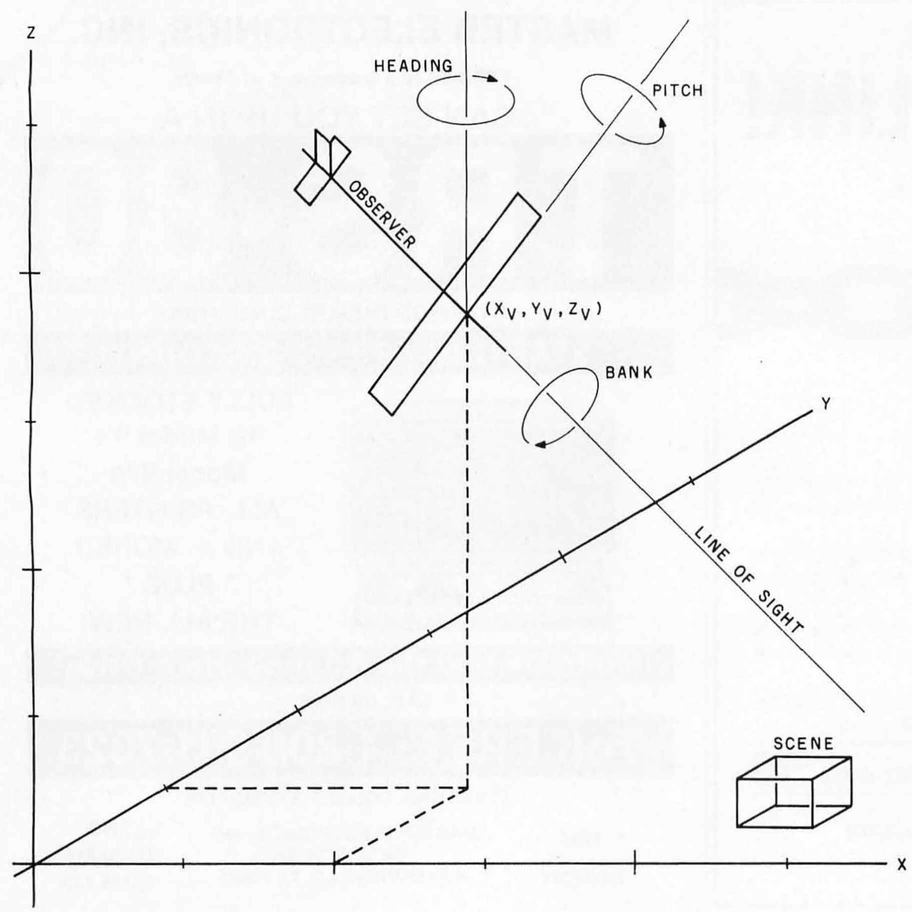

Figure 9: *Viewing parameters. In order to specify a unique view of an object in three dimensions, it is necessary to declare the observer's location and the angular position of the line of sight. The observer is at the point (X_V, Y_V, Z_V), and the line of sight is specified by the aircraft's pitch, bank, and heading.*

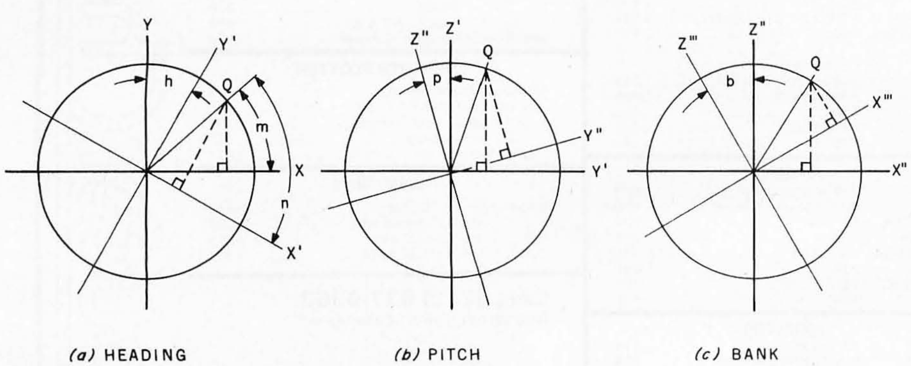

Figure 10: *Rotational transformation of a point. These figures illustrate the relationships between the coordinates of a point Q and the coordinates of Q in a Cartesian system where the observer is in the standard position. The standard position occurs when the observer is located at the origin, the line of sight is the positive y axis, and the "wings" lie on the x axis. In this position, the observer's pitch, bank, and heading are defined to be zero. See equations (1) through (9) in the text.*

the observer's view of the point, but simplifies the rotations that follow. Furthermore, a rotation of the coordinate axes will not affect the observer's perception of the point Q if the point's coordinates undergo an appropriate rotational transformation.

Since three rotations from the standard position determine a line of sight, three rotations of the coordinate axes are needed to bring the observer into the standard position of the rotated coordinate system. Again, the order of rotation is important. First, the x and y axes are rotated about the z axis by the amount of the heading, h, so that the airplane's fuselage is in the $y'z'$ plane (zero heading in the $x'y'z'$ system). Next, the y' and z' axes are rotated about the x' axis by the amount of the pitch, p, so that the fuselage lies on the y'' axis (zero heading and zero pitch in the $x''y''z''$ system). Finally, the x'' and z'' axes are rotated about the y'' axis by the amount of the bank, b, so that the pilot is in the standard position in the $x'''y'''z'''$ coordinate system (zero heading, zero pitch, and zero bank in the $x'''y'''z'''$ system).

With each rotation of the coordinate axes, the coordinates that specify any point will change. Figure 10 illustrates the relationships between the original coordinate system and the three different primed systems. Figure 10a shows that for any point $Q(X,Y,Z)$

$$\begin{aligned} X' &= R\cos(n) \\ &= R\cos(m+b) \\ &= R\cos(m)\cos(h) \\ &\quad + R\sin(m)\sin(h) \\ &= X\cos(h) - Y\sin(h) \end{aligned} \tag{1}$$

$$\begin{aligned} Y' &= R\sin(n) \\ &= R\sin(m+h) \\ &= R\sin(m)\cos(h) \\ &\quad + R\cos(m)\sin(h) \\ &= Y\cos(h) + X\sin(h) \end{aligned} \tag{2}$$

and

$$Z' = Z \tag{3}$$

We can see from figure 10b that

$$X'' = X' \tag{4}$$

$$Y'' = Y'\cos(p) + Z'\sin(p) \tag{5}$$

and

$$Z'' = Z'\cos(p) - Y'\sin(p) \qquad (6)$$

and we can see from figure 10c that

$$X''' = X''\cos(b) + Z''\sin(b) \qquad (7)$$

$$Y''' = Y'' \qquad (8)$$

and

$$Z''' = Z''\cos(b) - X''\sin(b) \qquad (9)$$

Substituting (1), (2), and (3) into (4), (5), and (6), and then substituting these results into (7), (8), and (9) yields

$$\begin{aligned} X''' = {} & [\cos(b)\cos(h) \\ & - \sin(h)\sin(p)\sin(b)]\,X \\ & + [-\cos(b)\sin(h) \\ & - \sin(p)\cos(h)\sin(b)]\,Y \\ & + [\cos(p)\sin(b)]\,Z \end{aligned}$$

$$\begin{aligned} Y''' = {} & [\sin(h)\cos(p)]\,X \\ & + [\cos(p)\cos(h)]\,Y \\ & + [\sin(p)]\,Z \end{aligned}$$

and

$$\begin{aligned} Z''' = {} & [-\cos(h)\sin(b) \\ & - \sin(h)\sin(p)\cos(b)]\,X \\ & + [\sin(h)\sin(b) \\ & - \sin(p)\cos(h)\cos(b)]\,Y \\ & + [\cos(p)\cos(b)]\,Z \qquad (10) \end{aligned}$$

These equations can be represented using matrix notation, as shown in figure 11. By multiplying out the three matrices, these equations relate the coordinates of a point Q in the xyz system to the coordinates of Q in a system, $x'''y'''z'''$, where the observer is in the standard position. Keep in mind that the observer is assumed to be at the origin in the xyz system prior to these calculations. Of course, the observer is also at the origin of the $x'''y'''z'''$ system because of the definition of standard position.

$$\begin{bmatrix} X''' \\ Y''' \\ Z''' \end{bmatrix} = \overset{\text{BANK}}{\begin{bmatrix} \cos(b) & 0 & \sin(b) \\ 0 & 1 & 0 \\ -\sin(b) & 0 & \cos(b) \end{bmatrix}} \overset{\text{PITCH}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(p) & \sin(p) \\ 0 & -\sin(p) & \cos(p) \end{bmatrix}} \overset{\text{HEADING}}{\begin{bmatrix} \cos(h) & -\sin(h) & 0 \\ \sin(h) & \cos(h) & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$= \begin{bmatrix} \cos(b)\cos(h) - \sin(h)\sin(p)\sin(b) & -\cos(b)\sin(h) - \sin(p)\cos(h)\sin(b) & \cos(p)\sin(b) \\ \sin(h)\cos(p) & \cos(p)\cos(h) & \sin(p) \\ -\cos(h)\sin(p) - \sin(h)\sin(p)\sin(b) & \sin(h)\sin(b) - \sin(p)\cos(h)\cos(b) & \cos(p)\cos(b) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 11: *Equation (10) represented using matrix notation.*

Projecting into the Picture Plane

Once the observer is in the standard position, it is easy to compute the perspective of a point. Remember that the perspective of a point Q is the intersection of the picture plane and the projector line joining the observer and the point. Since the standard position is in use, the observer is located at the origin and the line of sight is the positive y''' axis.

We have to define a new coordinate system for the picture plane. The two axes in the picture plane are labeled u and v such that the u axis is parallel to the x''' axis and the v axis is parallel to the z''' axis. Thus, the observer interprets the u axis to be

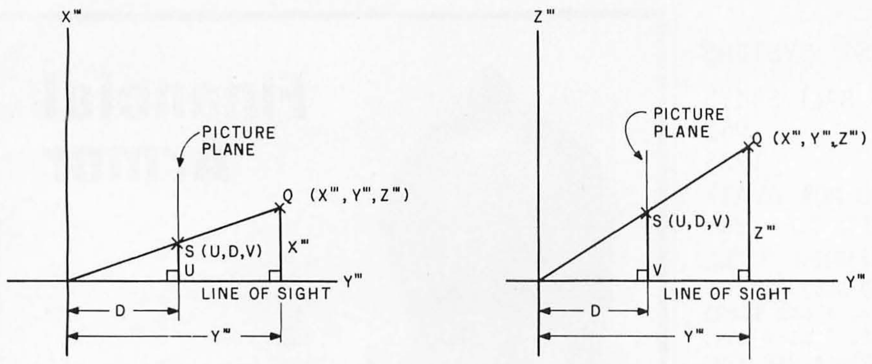

Figure 12: *Projecting a point. Once the observer is in the standard position, it is simple to compute the perspective projection of a point Q. The perspective projection of a point is the intersection of the point's projector and the picture plane. In this figure, D is the distance between the observer and the picture plane. See equations (11) through (14) in the text.*

horizontal and the v axis to be vertical. Figure 12 illustrates the relationship between the $x'''y'''z'''$ coordinates of a three-dimensional point P and the u-v coordinates of the point's projection in the picture plane. If D is the distance between the observer and the picture plane, from similar triangles

$$U = [D X''']/Y''' \tag{11}$$

and

$$V = [D Z''']/Y''' \tag{12}$$

When displaying a perspective drawing on a flat screen, it is often convenient to locate the intersection of the viewer's line of sight and the picture plane at the center of the screen. In this case the u and v axes are translated within the plane, and (11) and (12) take the form

$$U = \text{midh} + [\text{ppu}\, D X''']/Y''' \tag{13}$$

and

$$V = \text{midv} + [\text{ppu}\, D Z''']/Y''' \tag{14}$$

In these equations, (midh, midv) is the center of the viewing screen and ppu (pixels per unit) is a scaling factor to compensate for the fact that the unit of measurement of the screen is not the same as the unit of measurement in the coordinate axes.

It is often good to assume that the video monitor is the picture plane. In this case, the distance between the observer and the picture plane, D, is simply the distance between the computer operator and the terminal's screen. In this configuration, the viewing screen is truly a window through which to view the world. Since a perspective view of a three-dimensional line is always a two-dimensional line, we need only translate, rotate, and project two points for each edge of the scene. The line segment joining the two projected points will accurately represent the edge. However, if one of the endpoints of the segment lies outside the viewing screen, the line segment will have to undergo the process called "clipping." A clipping subroutine just ensures that only the part of the line segment that is within the boundaries of the screen will be drawn.

The Programs

Listing 1 and listing 2 are, respectively, BASIC and Pascal programs developed for the Apple II computer. The programs use the concepts discussed in this article to develop a perspective view of a three-dimensional object. In these programs, the observer can view the object, centered at the origin, from any point on a celestial sphere surrounding the object. The Pascal version includes capabilities for clipping and, in addition, for colored edges.

Figure 7 illustrates the data structure for both programs. In the BASIC version, the data is stored within the

Listing 1: *An Applesoft BASIC program that produces a perspective view of a three-dimensional object.*

```
50  REM ---WRITTEN BY ANDREW PICKHOLTZ---
70  REM ---JANUARY 1981---
100  HOME
120  VTAB 9: HTAB 3
140  PRINT "PERSPECTIVE VIEW OF A 3-D OBJECT"
160  PRINT : PRINT
180  PRINT "WRITTEN BY ANDREW PICKHOLTZ - JAN 1981"
200  VTAB 18
220  PRINT " PADDLE #0 CONTROLS THE VIEWER'S PITCH"
240  PRINT "PADDLE #1 CONTROLS THE VIEWER'S HEADING"
280  PRINT : PRINT : HTAB 6: FLASH
300  PRINT "WAIT WHILE LOADING VERTICES"
320  NORMAL : FOR SC = 1 TO 3000: NEXT SC
340 X = Y = Z = X3 = Y3 = Z3 = AM = BM = CM = DM = EM = FM = GM = HM =
 IM = D = P =  B
 = H = U = V = U1 = V1 = 0
360  DIM V(50,3),E(100)
370  REM ---READ DATA---
380  READ NV
400  FOR P = 1 TO NV
420  READ V(P,1),V(P,2),V(P,3)
440  NEXT P
460  READ NE
470  FOR E = 1 TO NE
480  READ E(E)
490  NEXT E
500  HOME : HGR : HCOLOR= 0: HPLOT 0,0
510  REM ---COMPUTE OBSERVER'S PARAMETERS---
520 D = 75
540 P = 6.28 *  PDL (0) / 255 - 3.1416
560 B = 0
580 H = 6.28 *  PDL (1) / 255
600  GOSUB 20100
620 XV =  - D * CP * SH: REM  ---SEE SUB.---
640 YV =  - D * CP * CH
660 ZV =  - D * SP
700  REM ---PROJECT NE POINTS---
720  FOR E = 1 TO NE
800 X = V( ABS (E(E)),1)
820 Y = V( ABS (E(E)),2)
840 Z = V( ABS (E(E)),3)
860  GOSUB 20720
900  IF E(E) > 0 THEN  HCOLOR= 3: HPLOT U1,V1 TO U,V
920 U1 = U:V1 = V
940  NEXT E
999  REM ---PREPARE FOR NEW FRAME---
1000  VTAB 21: HTAB 8: PRINT "PERSPECTIVE OF A BLOCK"
1010  PRINT "PROGRAM WRITTEN BY ANDREW PICKHOLTZ"
1020  HTAB 8: PRINT "PITCH="; INT (57.2 * P);"  HEADING="; INT (57.2 * H)
1060  PRINT "BUTTON #1 TO END-#0 FOR P=     , H=     ";
1070  VTAB 24: HTAB 27: PRINT "     , H=     ";
1080  VTAB 24: HTAB 27: PRINT  INT (360 *  PDL (0) / 255 - 180);
1100  VTAB 24: HTAB 35: PRINT  INT (360 *  PDL (1) / 255);
1110  REM ---CHECK FOR BUTTON PRESS---
1120  IF  PEEK ( - 16287) > 127 THEN 500
1140  IF  PEEK ( - 16286) < 128 THEN 1070
1160  TEXT
1180  END
20000  REM --- 3D PROJECTION SUBS. ---
20100  REM --- SET UP MATRIX ELEMENTS GIVEN PITCH, BANK, HEADING ---
20120  REM  --- USE TRANSCENDENTAL FUNCTIONS AS FEW TIMES AS POSSIBLE ---
20140 CH =  COS (H):SH =  SIN (H)
20160 CP =  COS (P):SP =  SIN (P)
20180 CB =  COS (B):SB =  SIN (B)
20200  REM  --- SET UP MATRIX ---
20220 AM = CB * CH - SH * SP * SB
20240 BM =  - CB * SH - SP * CH * SB
20260 CM = CP * SB
20280 DM = SH * CP
20300 EM = CP * CH
20320 FM = SP
20340 GM =  - CH * SB - SH * SP * CB
20360 HM = SH * SB - SP * CH * CB
20380 IM = CP * CB
20400  RETURN
```

Listing 1 continued on page 492

program. In the Pascal version, the data must be entered into a disk file. Listing 3 is a simple program that will store the data for the Pascal system.

The first element of data for both programs is the number of vertices that compose the object. Each vertex is then specified by its three coordinates. The unit of measurement in the coordinate system is the centimeter. If you use a monitor that does not measure 12 inches diagonally, you should change the ppu described in equations (13) and (14) to assure a truly accurate representation of distance. The programs assign a number to each vertex that is in the data file; the first vertex is number one, the second number two, and so on.

Following the vertices is the number of lines that are to be projected. This number also indicates how many numbers remain in the file. All of the remaining numbers refer to vertices that are to be projected. If the number is a positive integer, the programs draw a white line from the previously projected vertex to the vertex specified by the integer. The Pascal version will draw colored lines if you append a decimal point and a digit to the integer; point one specifies the color green, point two the color violet, and three and four specify orange and blue, respectively. A negative integer indicates that only the indicated vertex should be plotted, not a line as before.

Both programs have a subroutine (SETUP in the Pascal listing; line 20000 in the BASIC) that computes the elements of matrix in figure 11. You input the pitch and heading of the observer by using the Apple's paddles. The observer's bank is set to zero. Since the observer is on a celestial sphere of radius 75 centimeters, the location of the observer can be computed using a conversion from spherical to rectangular coordinates. Thus:

viewer's x coordinate
$$= -75\cos(p)\sin(h)$$
viewer's y coordinate
$$= -75\cos(p)\cos(h)$$
viewer's z coordinate
$$= -75\sin(p)$$

Listing 1 continued:

```
20600  REM --- SUB. TO TRANSFORM A 3D POINT TO A 2D ---
20620  REM --- X,Y,Z IS THE 3D POINT WHICH IS TRANSFORMED INTO U,V ---
20640  REM --- XV,YV,ZV ARE THE COORDINATES OF THE VIEWER ---
20660  REM --- D IS THE DISTANCE BETWEEN THE OPERATOR'S EYE AND THE SCREEN IN CM
---
20680  REM --- GOSUB 20100 EVERY TIME THE VIEWERS PITCH, BANK, OR HEADING CHANGES
---
20700  REM --- TRANSLATE SO THAT VIEWER IS AT THE ORIGIN ---
20720 X = X - XV
20740 Y = Y - YV
20760 Z = Z - ZV
20780  REM --- ROTATE SO THAT THE VIEWER IS LOOKING DOWN THE Y-AXIS ---
20800 X3 = AM * X + BM * Y + CM * Z
20820 Y3 = DM * X + EM * Y + FM * Z
20840 Z3 = GM * X + HM * Y + IM * Z
20860  REM ---PROJECT INTO 2D SCREEN---
20880 U = 135 + 13.5 * D * X3 / Y3
20900 V = 80 - 11.5 * D * Z3 / Y3
20920  RETURN
30000  REM  ---NUMBER OF VERTICES---
30020  DATA  8
30040  REM  ---VERTEX COORDINATES---
30060  DATA  5.25,-2,3.25
30080  DATA  -5.25,-2,3.25
30100  DATA  -5.25,-2,-3.25
30120  DATA  5.25,-2,-3.25
30140  DATA  5.25,2,3.25
30160  DATA  -5.25,2,3.25
30180  DATA  -5.25,2,-3.25
30200  DATA  5.25,2,-3.25
31000  REM  --- NUMBER OF EDGES---
31020  DATA  16
31040  REM  ---- EDGES ---
31060  REM ---NEG. EDGES START NEW CURVE---
31080  DATA  -1,2,3,4,1,5,6,7,8,5
31100  DATA  -2,6,-3,7,-4,8
```

Note that the distance between the observer and the picture plane (DIS or D) is equal to the distance between the observer and the center of the object (the radius of the sphere). Therefore, the dimensions of the perspective of the object will approximate those of the object itself.

The second subroutine in each program (PROJECT, or line 20600) does the computations needed to project each point. Thus, each subroutine first transforms the point into the triple-primed system. Then, each projects the point into the picture plane using formulas (13) and (14). The program has to call this second subroutine every time a point is projected. The program has to call the first subroutine only when the viewing parameters change.

Using the Programs

The BASIC program in listing 1 is ready to run as is. Lines 30000-31100 of listing 1 contain the data for a rect-

Text continued on page 500

Listing 2: *Object3d, an Apple Pascal program that produces a perspective view of a three-dimensional object.*

```
(*$S+*)  (* SWAPPING OPTION *)

PROGRAM OBJECT3D;

(* WRITTEN BY ANDREW PICKHOLTZ - JANUARY 1981 *)
(* PROJECT THE IMAGE OF A 3-D OBJECT INTO THE SCREEN *)

USES TURTLEGRAPHICS, APPLESTUFF, TRANSCEND;

CONST
  MIDH = 135;   (* SCREEN CENTER *)
  MIDV = 95;
  PPCMH = 13.5; (* POINTS PER CM *)
  PPCMV = 11.5;
  MAXVER = 300;
  MAXEDG = 200;
  PI = 3.1416;

TYPE
  RANGVER = 1..MAXVER;
  RANGEDG = 1..MAXEDG;
  POINT3D = ARRAY [(X,Y,Z)] OF REAL;
  POINT2D = ARRAY [(U,V)] OF REAL;
  MANY3D = ARRAY [RANGVER] OF POINT3D;


VAR
  EDGE : ARRAY [RANGEDG] OF REAL;
  VERTEX : MANY3D;
  DATAFILE : FILE OF REAL;
  OBJCTNAME : STRING;
  OBSCOOR : POINT3D; (*OBSERVER'S COORDINATES*)
  P,B,H : REAL;
  SP,CP,SB,CB,SH,CH : REAL;
  AM,BM,CM,DM,EM,FM,GM,HM,IM : REAL;
  NUMVER : RANGVER;
  NUMEDG : RANGEDG;
  DIS : REAL;
  DONE : BOOLEAN;

PROCEDURE TITLE;
BEGIN
  PAGE (OUTPUT);
  GOTOXY (3,6);
  WRITELN ('PERSPECTIVE VIEW OF A 3-D OBJECT');
  WRITELN; WRITELN;
  WRITELN ('WRITTEN BY ANDREW PICKHOLTZ - JAN 1981');
  GOTOXY (1,16);
  WRITELN ('PADDLE #0 CONTROLS OBSERVER''S PITCH');
  WRITELN ('PADDLE #1 CONTROLS OBSERVER''S HEADING');
  WRITELN;
  WRITE ('OBJECT (FILE) TO BE DISPLAYED? ');
END;


PROCEDURE READDATA;
VAR I : RANGVER;
  J : RANGEDG;
FUNCTION LOAD:REAL;
BEGIN
  LOAD := DATAFILE↑;
  GET (DATAFILE);
END;
BEGIN
  READLN (OBJCTNAME);
  RESET (DATAFILE,OBJCTNAME);
         (* LOAD VERTICES *)
  NUMVER := TRUNC (LOAD);
  FOR I:= 1 TO NUMVER DO
  BEGIN
    VERTEX [I,X] := LOAD;
```

Listing 2 continued on page 496

Listing 2 continued:

```
    VERTEX [I,Y] := LOAD;
    VERTEX [I,Z] := LOAD;
  END;
         (* LOAD EDGES *)
  NUMEDG := TRUNC (LOAD);
  FOR J:= 1 TO NUMEDG DO
  BEGIN
    EDGE [J] := LOAD;
  END;
  CLOSE (DATAFILE);
END; (*READDATA*)

PROCEDURE SETUP;
CONST EPSILON = 2.5E-2;
BEGIN
      (* PITCH, BANK, HEADING, DISTANCE *)

  DIS:=75;
  P:=2*PI*PADDLE(0)/255-PI;
  IF ABS(P) < EPSILON THEN P:=0;
  B:=0;
  H:=2*PI*PADDLE(1)/255;
     (* TRANSCENDENTAL FUNCTIONS *)
  SP:=SIN(P); CP:=COS(P);
  SB:=SIN(B); CB:=COS(B);
  SH:=SIN(H); CH:=COS(H);
     (* MATRIX COMPONENTS *)
  AM := CH*CB-SH*SP*SB;
  BM := -SH*CB-CH*SP*SB;
  CM := CP*SB;
  DM := SH*CP;
  EM := CP*CH;
  FM := SP;
  GM := -CH*SB-SH*SP*CB;
  HM := SH*SB-CH*SP*CB;
  IM := CP*CB;
     (* OBSERVER'S COORDINATES *)
  OBSCOOR[X] := -DIS*CP*SH;
  OBSCOOR[Y] := -DIS*CP*CH;
  OBSCOOR[Z] := -DIS*SP;
END;

PROCEDURE PROJECT (PT3D : POINT3D;VAR PT2D : POINT2D);
VAR ROTPT : POINT3D;
BEGIN
        (* TRANSLATE SO THAT OBSERVER IS AT ORIGIN *)
  PT3D[X] := PT3D[X] - OBSCOOR[X];
  PT3D[Y] := PT3D[Y] - OBSCOOR[Y];
  PT3D[Z] := PT3D[Z] - OBSCOOR[Z];
        (* ROTATE SO THAT OBSERVER IS LOOKING DOWN Y-AXIS *)
  ROTPT[X] := AM*PT3D[X] + BM*PT3D[Y] + CM*PT3D[Z];
  ROTPT[Y] := DM*PT3D[X] + EM*PT3D[Y] + FM*PT3D[Z];
  ROTPT[Z] := GM*PT3D[X] + HM*PT3D[Y] + IM*PT3D[Z];
        (* PROJECT INTO PICTURE PLANE AT DISTANCE DIS *)
  PT2D [U] := MIDH + PPCMH*DIS*ROTPT[X]/ROTPT[Y];
  PT2D [V] := MIDV + PPCMV*DIS*ROTPT[Z]/ROTPT[Y];
END;

PROCEDURE DRAW;
VAR
  J : RANGEDG;
  SCRNPT : POINT2D;
BEGIN
  SETUP;
  FOR J := 1 TO NUMEDG DO
  BEGIN
          (* DELETE FRACTIONAL PART AND PROJECT *)
    PROJECT ( VERTEX[ABS(TRUNC(EDGE [J]))], SCRNPT );
          (* IF EDGE IS NEG THEN LINE IS NOT DISPLAYED *)
    IF EDGE [J] < 0
     THEN PENCOLOR (NONE)
          (* FRACTIONAL PART OF EDGE DETERMINES COLOR *)
     ELSE
     CASE TRUNC(10*(EDGE [J]-TRUNC(EDGE [J]))) OF
       0 : PENCOLOR (WHITE);
       1 : PENCOLOR (GREEN);
       2 : PENCOLOR (VIOLET);
       3 : PENCOLOR (ORANGE);
       4 : PENCOLOR (BLUE);
     END;
     MOVETO (ROUND(SCRNPT[U]), ROUND(SCRNPT[V]));
  END;
END; (*DRAW*)
```

Listing 2 continued on page 498

Listing 2 continued:

```
PROCEDURE NEWFRAME;
CONST  RADDEG = 57.2;
VAR ADVANCE : BOOLEAN;
       S : STRING;
       PDL : REAL;
BEGIN
       (* COMMENTS AROUND OBJECT *)
  PENCOLOR (NONE);
  MOVETO (15,183);
  WSTRING ('PERSPECTIVE VIEW OF A ');
  WSTRING (OBJCTNAME);
  MOVETO (10,174);
  STR (ROUND (P*RADDEG),S);
  WSTRING ('PITCH = ');
  WSTRING (S);
  MOVETO (180,174);
  STR (ROUND (H*RADDEG),S);
  WSTRING ('HEADING = ');
  WSTRING (S);
  MOVETO (10,2);
  WSTRING ('BUTTON #1 ENDS - #0 FOR ');
       (* UPDATE COMMENTS - CHECK BUTTONS *)
  REPEAT
    MOVETO (178,2);
    WSTRING ('            ');
    MOVETO (178,2);
    PDL := PADDLE(0)/255*360-180;
    STR (ROUND (PDL),S);
    WSTRING ('P=');
    WSTRING (S);
    MOVETO (227,2);
    PDL := PADDLE(1)/255*360;
    STR (ROUND (PDL),S);
    WSTRING ('H=');
    WSTRING (S);
    ADVANCE := BUTTON (0);
    DONE := BUTTON (1);
  UNTIL ADVANCE OR DONE;
  FILLSCREEN (BLACK);
END;
```

```
BEGIN
  TITLE;
  READDATA;
  INITTURTLE;
  REPEAT
    DRAW;
    NEWFRAME;
  UNTIL DONE;
  PAGE (OUTPUT);
  TEXTMODE;
  WRITELN;
  WRITELN ('DONE...');
END.
```

Listing 3: *A Pascal program that stores data for use by Object3d, the program given in listing 2.*

```
PROGRAM FILEWRITE;

(* PROGRAM TO INSERT DATA INTO A FILE *)

VAR
  FILENAME : STRING;
  DISK : FILE OF REAL;
  NUMBER : REAL;

BEGIN
  WRITELN;
  WRITE ('NAME OF OBJECT (FILE)? ');
  READLN (FILENAME);
  WRITELN ('< CNTL-C >  TO STOP');
  WRITELN;
  REWRITE (DISK,FILENAME);
  REPEAT
    READLN (NUMBER);
    DISK↑ := NUMBER;
    PUT (DISK);
  UNTIL EOF;
  CLOSE (DISK,LOCK);
END.
```

Text continued from page 492:

Listing 4: *Changes for the BASIC program in listing 1 that will produce a perspective of a dodecahedron.*

```
]LIST

430 V(P,1) = 6 * V(P,1):V(P,2) = 6 * V(P,2):V(P,3) = 6 * V(P,3)
1000  VTAB 21: HTAB 4: PRINT "PERSPECTIVE OF A DODECAHEDRON"
30000  REM  ---NUMBER OF VERTICES---
30020  DATA  20
30040  REM  ---VERTEX COORDINATES---
30060  DATA  0,-.3568,.9342,.5774,-.5774,.5774
30080  DATA  .9342,0,.3568,.5774,.5774,.5774
30100  DATA  0,.3568,.9342,-.5774,.5774,.5774
30120  DATA  -.9342,0,.3568,-.5774,-.5774,.5774
30140  DATA  -.3568,-.9342,0,.3568,-.9342,0
30160  DATA  .3568,.9342,0,-.3568,.9342,0
30180  DATA  -.5774,-.5774,-.5774,0,-.3568,-.9342
30200  DATA  .5774,-.5774,-.5774,.9342,0,-.3568
30220  DATA  .5774,.5774,-.5774,0,.3568,-.9342
30240  DATA  -.5774,.5774,-.5774,-.9342,0,-.3568
31000  REM  --- NUMBER OF EDGES---
31020  DATA  40
31040  REM  --- EDGES ---
31060  REM ---NEG. EDGES START NEW CURVE---
31080  DATA  -1,2,3,4,5,6,7,8,1,5
31100  DATA  -14,15,16,17,18,19,20,13,14,18
31120  DATA  -8,9,10,2
31140  DATA  -13,9
31160  DATA  -15,10
31180  DATA  -4,11,12,6
31200  DATA  -17,11
31220  DATA  -19,12
31240  DATA  -16,3
31260  DATA  -7,20
31280  END
```

Listing 5: *A Pascal program that displays an accurately proportioned model of a DNA molecule.*

```
PROGRAM MAKEDNA;

(* PROGRAM TO COMPUTE DOUBLE-HELIX *)
(* ANDREW PICKHOLTZ - JAN 1981 *)

USES APPLESTUFF,TRANSCEND;

CONST
       (* DESCRIPTION OF DOUBLE-HELIX *)
       (* SEE J.D.WATSON, MOLECULAR BIOLOGY OF THE GENE *)
       (* SECOND EDITION, PAGES 261-262 *)
       (* ALL UNITS IN ANGSTROMS *)
 RADIUS = 10;
 HTURN = 34;      (* HEIGHT OF ONE TURN *)
 NNPT = 10;       (* NUMBER OF NUCLEOTIDES PER TURN *)
 OFFSET = 2.402; (* OFFSET FOR SECOND HELIX IN RADIANS *)
 PI = 3.1416;
 CMPA = 0.1569;  (* CM PER ANGSTROM FOR MAGNIFICATION *)
 NNUCL = 15;     (* HALF NUMBER OF NUCLEOTIDES TO DISPLAY *)

VAR
 NUCL : -NNUCL..NNUCL;
 I,J : INTEGER;
 DISK : FILE OF REAL;
 COLOR : REAL;
```

Listing 5 continued on page 502

angular parallelepiped (a prism whose bases are parallelograms), i.e., a block. Listing 4 has the changes that should be made to project a dodecahedron (a solid with 12 faces) instead of a block. To produce a full-screen dodecahedron, you must multiply the data elements in lines 30060-30240 by 6. Once compiled, the Pascal program in listing 2 will display any object that is represented in a disk file. The program in listing 3 will load data into a file.

Objects can also be created by the computer. For example, listing 5 contains a Pascal program that will create an accurately proportioned model of a DNA molecule. The double helix is represented by 62 vertices connected by more than 100 straight lines.

Photo 1 is an actual photograph of a real dodecahedron. Photos 2 through 6 show examples of the use of the programs. Photo 2 displays computer-drawn perspectives from various viewing positions. Similarly, photo 3 shows the wire-frame perspective of a block. Photo 4 shows drawings of an object that you are more likely to encounter, a house. You can see that these photos exhibit the properties of true perspective that were shown in figure 4. The DNA molecule shown in photo 5 is even more interesting. The orientation of the double helix displayed in photo 5a is like those found in many textbooks.

Pascal draws an object significantly faster than does BASIC. On the system used, however, the speed at which the object is drawn and the screen erased is not nearly fast enough to produce the appearance of smooth motion, let alone freedom from flickering (this would require a minimum of 30 frames per second). Thus, although the programs have the inherent capability of displaying a continuous change in view of the object, either a faster machine or a more efficient compiler, or both, would be necessary to achieve this result.

Extensions, Anyone?

The most obvious extension to wire-frame perspective is to remove from the display the lines and sur-

Listing 5 continued:

```
PROCEDURE DUMP (DATA : REAL);
BEGIN
  DISK↑ := DATA;
  PUT (DISK);
END;
BEGIN
  WRITELN;
  WRITELN ('WRITING');
  REWRITE (DISK,'DOUBLE-HELIX');
          (* NUMBER OF POINTS *)
  DUMP (2*(2*NNUCL+1));
  FOR NUCL := -NNUCL TO NNUCL DO
  BEGIN
          (* FIRST HELIX *)
    DUMP (CMPA*RADIUS* COS(NUCL*2*PI/NNPT));
    DUMP (CMPA*RADIUS* SIN(NUCL*2*PI/NNPT));
    DUMP (CMPA*HTURN/NNPT*NUCL);
          (* SECOND HELIX *)
    DUMP (CMPA*RADIUS* COS(NUCL*2*PI/NNPT+OFFSET));
    DUMP (CMPA*RADIUS* SIN(NUCL*2*PI/NNPT+OFFSET));
    DUMP (CMPA*HTURN/NNPT*NUCL);
  END;
          (* NUMBER OF EDGES *)
  DUMP (8*NNUCL);
  FOR J := 1 TO 2*NNUCL DO
  BEGIN
    I := 2*J-1;
    DUMP (-I);
    DUMP (I+2);
    COLOR := ((RANDOM MOD 4)+1)/10;
    DUMP (I+3+COLOR);
    DUMP (I+1);
  END;
  CLOSE (DISK,LOCK);
  WRITELN ('DONE...');
END.
```

faces that should be hidden, as shown in photo 6. A first step to accomplish this task of hidden-line removal is to define the faces of the object. We can do that by discarding the data structure represented in figure 7 in favor of the data structure in figure 6. The next step in hidden-line removal is to determine which faces shield the others.

Many methods of hidden-line removal are available. All require much more computer time than the wire-frame representation. One procedure, the depth-buffer algorithm, requires that the depth of every pixel on the screen be recorded. Before drawing a new point into the screen, the depth of the point to be displayed is compared with the depth of the existing pixel. The new point will be drawn only if it is closer to the observer than the existing screen point. Although this algorithm is relatively simple, it requires an enormous amount of computer memory.

Another method of hidden-line removal, the priority algorithm, requires that all the faces be sorted in the order that they are to be drawn into the screen. Thus, the faces in the foreground will block the faces behind them, since the foreground faces will be drawn last. One drawback of the priority algorithm is that it cannot draw cyclically overlapping polygons.

Wire-frame perspective could also

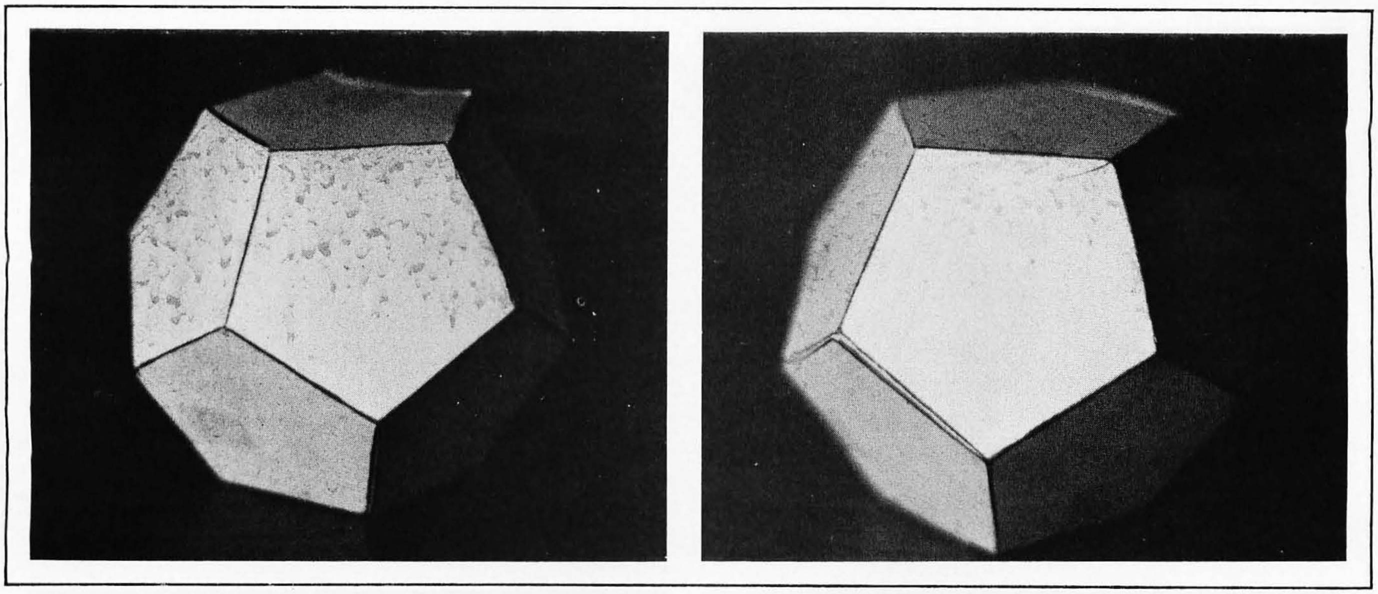

Photo 1: *A real regular dodecahedron.*

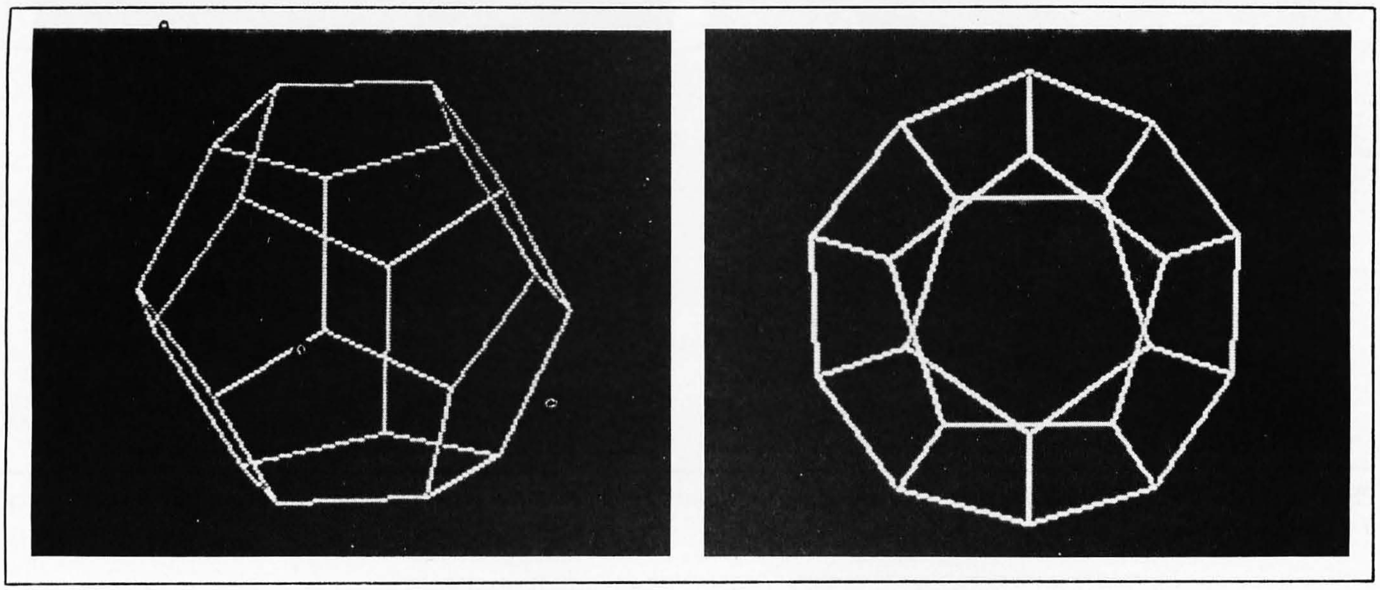

Photo 2: *A screen representation of a wire-frame dodecahedron. The video monitor was connected to an Apple II.*

be extended to draw curves rather than just straight lines. The Bezier and B-spline methods are among the many techniques for interpolating curves with a finite number of points.

The interposition of two perspective views of an object from slightly different positions can create the effect of stereo vision. One view could be drawn on the left-hand side of the screen and the other on the right-hand side. Alternatively, an anaglyph (a stereoscopic picture using two different colors, similar to three-dimensional movies now being shown on TV) can be produced by drawing one view in one color and the other slightly displaced view in another color. Of course, viewing glasses with correspondingly colored filters would be required to perceive the stereo effect of the anaglyph.

Many more difficult problems present themselves when searching for further extensions to the wire-frame perspective. Extraordinary realism in three-dimensional graphics can be achieved by including more of the physical characteristics of real objects, such as shading, shadowing, texture, reflectivity, and transparency. These characteristics could all improve the realism of a computer-drawn scene. Although an artist could produce a painting with all these characteristics, the computer-drawn scene could be manipulated interactively to present various alterations to the observer, such as different viewing angles, changes in scale, and even topological transformations of the scene.

Conclusion

The new generation of microcomputers that is now entering the marketplace will provide an abundance of opportunities for writing and viewing computer graphics. More powerful processors, higher-resolution monitors, and greater

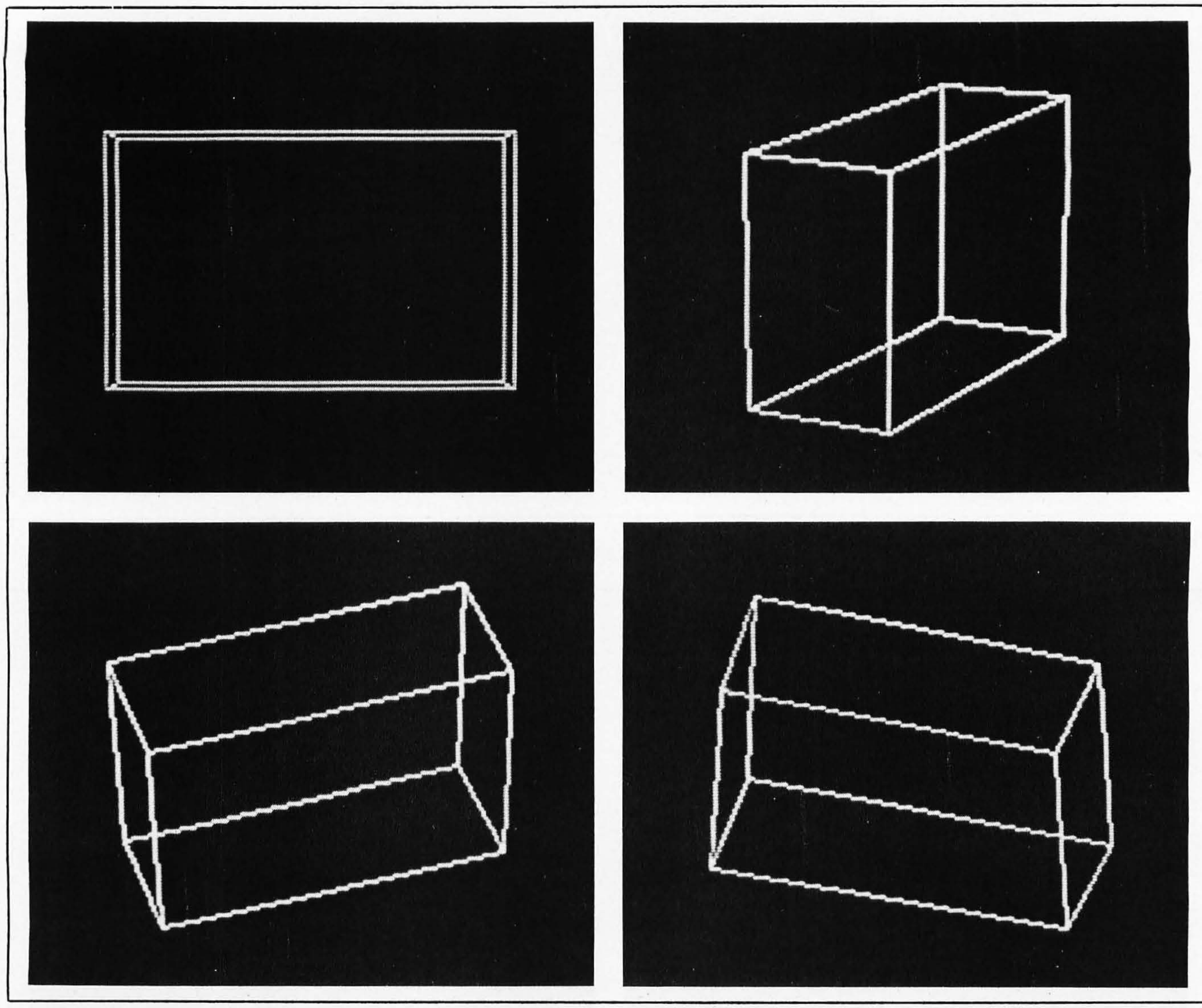

Photo 3: *Perspective view of a block from a variety of angles. Negative pitch produces views from above, and positive pitch produces a view from below.*

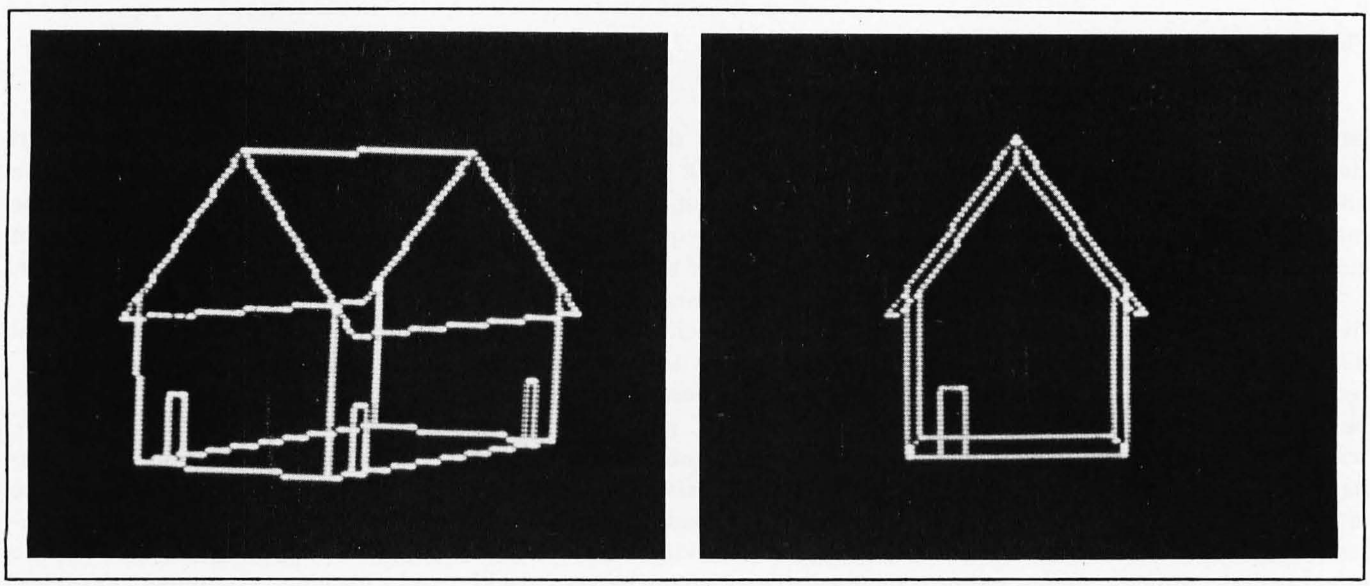

Photo 4: *Perspective view of a wire-frame house.*

memory-addressing capacity will enable programmers to use some of the techniques that were impossible on an Apple II. The same improvements will make computer graphics more exciting to both sophisticated and naive observers. I hope that this article not only interests today's Apple II owners, but also encourages them and others to write software that exploits the impressive graphics capabilities of the new machines. ■

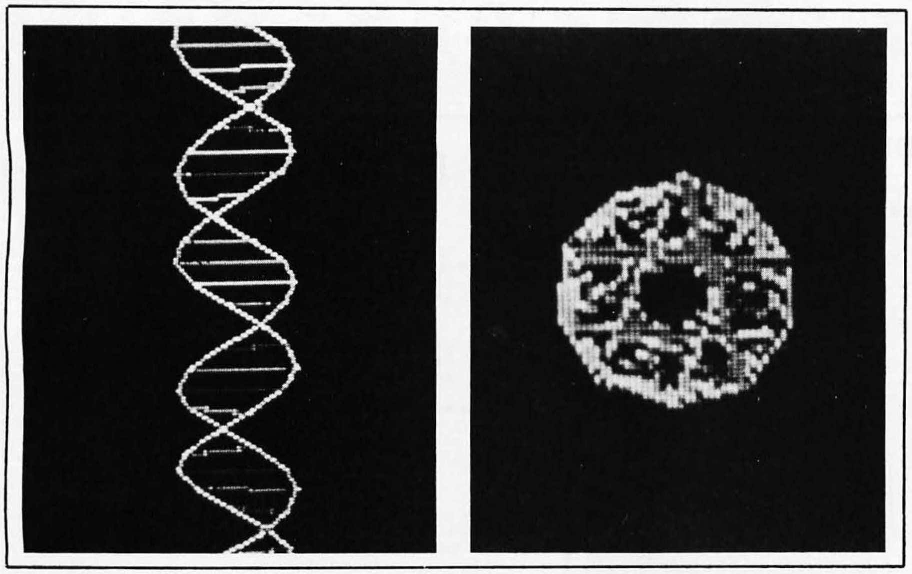

Photo 5: *Perspective view of a double helix. The observer's line of sight is perpendicular to the axis of the double helix, and then down the axis of the double helix.*

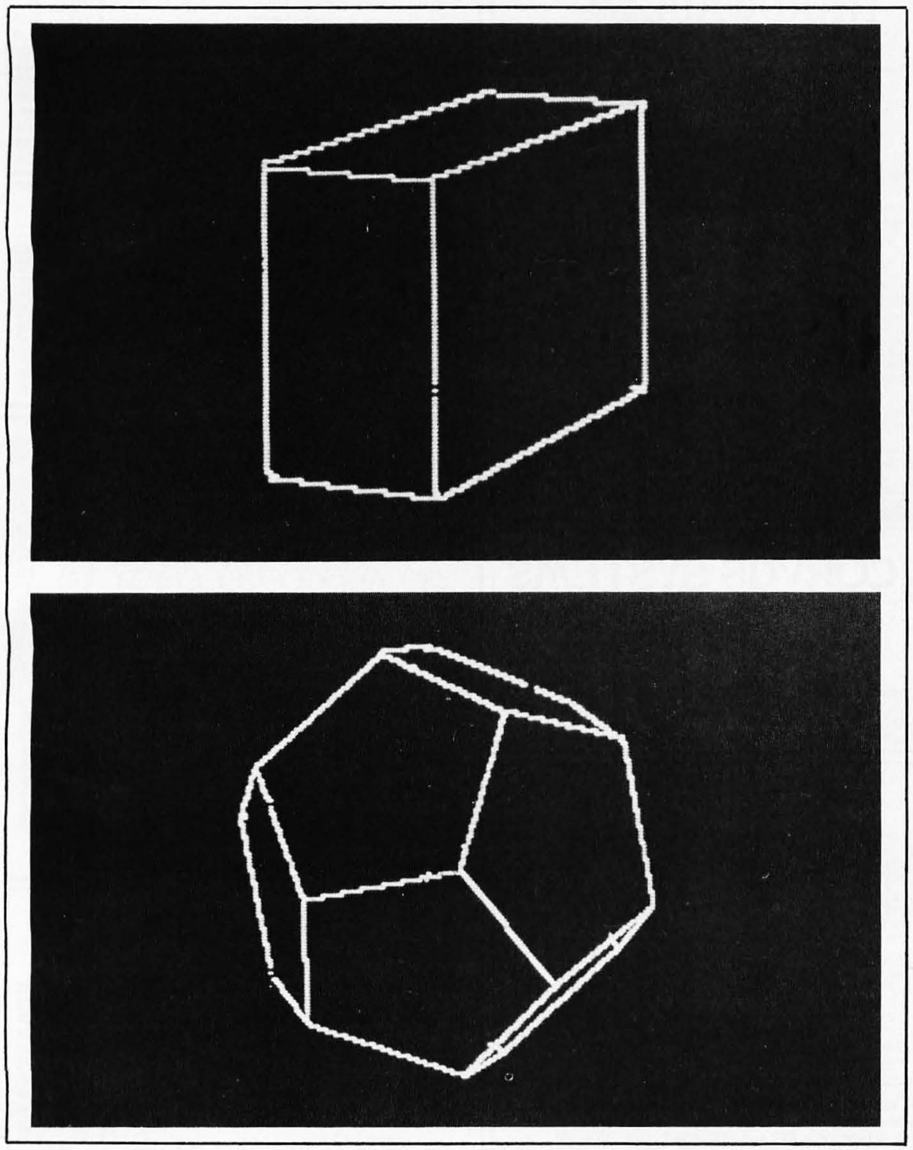

Photo 6: *Perspective of a block and a dodecahedron with lines that would be hidden by opaque surfaces removed from the representation (photos were manually produced).*

References

1. Albert, A. Adrian. *Solid Analytic Geometry*. Chicago: University of Chicago Press, 1966.
2. Artwick, Bruce A. *Three Dimensional Microcomputer Graphics*. Culver City, CA: Sublogic Company, 1977.
3. Chasen, Sylvan H. *Geometric Principles and Procedures for Computer Graphics Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
4. Foley, James and Andries van Dam. *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley, 1981.
5. Giloi, Wolfgang K. *Interactive Computer Graphics: Data Structures, Algorithms, Languages*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
6. Greco, Simon. *The Art of Perspective Drawing*. New York: M. Grumbacher, 1975.
7. Horn, Berthold K. P. "Hill Shading and the Reflectance Map." *Proceedings of the IEEE*, January 1981.
8. Hungerford, Joel C. "Graphic Manipulations Using Matrices." BYTE, September 1978, page 156.
9. *Image, Object, and Illusion* (A *Scientific American* Publication). San Francisco: W. H. Freeman and Company, 1974.
10. Langridge, Robert, Thomas E. Ferrin, Irwin D. Kuntz, and Michael L. Connolly. "Real-Time Color Graphics in Studies of Molecular Interactions." *Science*, February 13, 1981.
11. Lerner, Eric J. "The Computer Graphics Revolution." *Spectrum*, February 1981.
12. Newman, William M. and Robert F. Sproull. *Principles of Interactive Computer Graphics*, second ed. New York: McGraw-Hill, 1979.
13. Pearce, Peter and Susan Pearce. *Polyhedra Primer*. New York: Van Nostrand Reinhold, 1978.
14. Potsdamer, Jeffrey L. "The Mathematics of Computer Graphics." BYTE, September 1978, page 22.
15. Requicha, Aristides A. G. "Representations for Rigid Solids: Theory, Methods, and Systems." *ACM Computing Surveys*, December 1980.